Eine empirische Studie zur strukturellen Komplexität konzeptioneller Modelle – Grundlegung eines effizienten Ansatzes zur strukturellen Modellanalyse

Dominic Breuker, Hanns-Alexander Dietrich,
Johannes Püster, Matthias Steinhorst,
Jörg Becker, Patrick Delfmann

Veröffentlicht in:
Multikonferenz Wirtschaftsinformatik 2012
Tagungsband der MKWI 2012
Hrsg.: Dirk Christian Mattfeld; Susanne Robra-Bissantz

Braunschweig: Institut für Wirtschaftsinformatik, 2012

Eine empirische Studie zur strukturellen Komplexität konzeptioneller Modelle – Grundlegung eines effizienten Ansatzes zur strukturellen Modellanalyse

Dominic Breuker, Hanns-Alexander Dietrich, Johannes Püster, Matthias Steinhorst, Jörg Becker, Patrick Delfmann

Westfälische Wilhelms-Universität Münster,
European Research Center for Information Systems, 48149 Münster,
E-Mail: <Nachname>@ercis.uni-muenster.de

Abstract

Für die Analyse konzeptioneller Informationsmodelle existiert eine Vielzahl von Verfahren, die für eine bestimmte Modellierungssprache oder Anwendungsdomäne entwickelt wurden und abgegrenzte Analysezwecke unterstützen. Häufig besteht eine Teilaufgabe der Analyseverfahren in der Identifikation struktureller Modellmuster. Für dieses als Subgraphisomorphie bekannte Problem existieren effiziente, allgemein anwendbare Algorithmen, die bestimmte topologische Eigenschaften der zugrundeliegenden Graphen voraussetzen. Dieser Beitrag präsentiert die Ergebnisse der Untersuchung von 3000 Informationsmodellen und demonstriert, dass die meisten dieser Modelle entsprechende Eigenschaften aufweisen. Er präsentiert eine empirische Studie für die Entwicklung einer allgemein anwendbaren Analysemethode, die in viele Spezialentwicklungen integriert werden kann.

1 Motivation

In der Literatur der Wirtschaftsinformatik wird eine Vielzahl von Modellanalyseverfahren diskutiert, die spezielle Analyseprobleme in bestimmten Anwendungsdomänen und für bestimmte Modellierungssprachen lösen. In der Domäne des Geschäftsprozessmanagements können z. B. Verfahren zur Syntaxprüfung (z. B. [22]), zum Modellvergleich (z. B. [11]) oder zur Integration (z. B. [10]) von Prozessmodellen unterschieden werden. Andere Autoren stellen Verfahren vor, die Schwachstellen innerhalb von Prozessmodellen identifizieren (z. B. [4]) oder die die Einhaltung von Compliance-Regeln überprüfen (z. B. [20]).

Ein häufig auftretendes Unterproblem dieser Verfahren ist Identifikation struktureller Modellmuster. Da diese in der Regel als benannte, getypte Graphen dargestellt werden, verfolgt dieser Beitrag das Ziel, zu untersuchen, ob effiziente, aus der Graphentheorie bekannte Algorithmen zur Mustererkennung in konzeptionellen Informationsmodellen wiederverwendet

werden können. Die Laufzeitkomplexität eines Algorithmus, der das als Subgraphisomorphie bezeichnete Problem (vgl. [34]) der Mustererkennung im allgemeinen Fall löst, ist exponentiell (vgl. [8]). Aus diesem Grund liefert ein solcher Algorithmus in der Praxis bereits bei mittelgroßen Graphen keine Lösung in zufriedenstellender Laufzeit. Dem gegenüber stehen Modelle in der Praxis, die – insbesondere bei Großunternehmen und Konzernen, die im Vergleich zu eher kleinen Unternehmen besonders häufig konzeptionelle Modelle pflegen – die mehrere Tausend Knoten umfassen können. Eine breite Anwendbarkeit ist deshalb in der konzeptionellen Modellierung kaum möglich. Effiziente Algorithmen mit polynomieller Laufzeit sind nur für Graphen bekannt, die *planar* sind oder eine *begrenzte Baumweite* aufweisen (vgl. z. B. [13,14,19]). Es wird vermutet, dass konzeptionelle Modelle der Wirtschaftsinformatik zum großen Teil genau diese Eigenschaften besitzen. Damit wäre es möglich, das im Rahmen der Modellanalyse häufig auftretende Problem der Mustererkennung mit einem effizienten und gleichzeitig universell anwendbaren Verfahren zu unterstützen. Bisher ist dies an der hohen Laufzeitkomplexität der Subgraphisomorphie gescheitert. In der Literatur der Wirtschaftsinformatik wird daher eine Reihe von Spezialansätzen diskutiert, die die Komplexität des allgemeinen Problems durch Ausnutzung von besonderen Aspekten der zu Grunde liegenden Modellierungssprachen und des Anwendungszwecks umgehen. Es ist das Ziel dieses Beitrages zu untersuchen, inwiefern konzeptionelle Informationsmodelle die strukturellen Eigenschaften (Planarität und begrenzte Baumweite) erfüllen, die eine Integration effizienter Mustererkennungsalgorithmen in bestehende Analyseansätze ermöglichen. Dies birgt den Vorteil, dass die (potenziell aufwändige) Mustererkennungs-komponente dieser Ansätze durch allgemein anwendbare, mathematisch verifizierte und effiziente Verfahren ersetzt werden kann. Die Entwickler dieser speziellen Ansätze können sich so auf die analysezweckspezifische Konfiguration und Erweiterung dieser allgemeinen Verfahren konzentrieren.

Um zu zeigen, dass konzeptionelle Modelle über die graphentheoretischen Eigenschaften verfügen, die für eine effiziente Mustersuche benötigt werden, präsentiert der vorliegende Beitrag die Ergebnisse der Untersuchung von ca. 3000 Modellen. Das Ergebnis verdeutlicht, dass die große Mehrheit dieser Modelle tatsächlich planar ist bzw. eine beschränkte, geringe Baumweite aufweist. Der Beitrag legt somit die Grundlage für die Entwicklung eines generischen Ansatzes zur effizienten strukturellen Modellanalyse. Dieser Ansatz kann in viele, in der Literatur diskutierten Analyseverfahren integriert werden, um das häufig auftretende Problem der Mustererkennung effizient zu lösen. Wir beschränken uns auf die Identifikation struktureller Muster in konzeptionellen Modellen. Verhaltensmuster werden nicht betrachtet.

Der Artikel ist wie folgt strukturiert: Kapitel 2 diskutiert verwandte Arbeiten, die das Problem der Mustererkennung in Graphen (als Teilkomponente) beinhalten. In Kapitel 3 werden konzeptionelle Modelle graphentheoretisch formalisiert. Darüber hinaus führt dieses Kapitel das Problem der Mustererkennung in Graphen formal ein. Kapitel 4 führt die graphentheoretischen Eigenschaften *Baumweite* und *Planarität* ein, die für eine effiziente Mustererkennung benötigt werden. Kapitel 5 beschreibt die Datenbasis, auf der die empirische Studie durchgeführt wurde. Die Ergebnisse werden in Kapitel 6 vorgestellt. Der Beitrag schließt mit einer Zusammenfassung und einem Ausblick auf weiteren Forschungsbedarf in Kapitel 7.

2 Verwandte Arbeiten

Eine Vielzahl von Verfahren zur Analyse von Modellen basiert direkt oder indirekt auf dem Problem der Mustersuche. Im Folgenden werden einige solcher Ansätze aus den Bereichen *Compliance-Überprüfung*, *Identifikation von Schwachstellen in Geschäftsprozessen*, *Komplexitätsmanagement*, *Modellvergleich* und *Modelltransformation* vorgestellt.

Gegenstand der *Compliance-Überprüfung* ist die Kontrolle eines Modells auf Kompatibilität mit gesetzlichen oder regulatorischen Anforderungen. Diese können als abstraktes Muster formuliert werden. Dabei kann es sich einerseits um ein Positiv-Muster handeln, welches in gesetzeskonformen Modellen vorhanden sein muss, andererseits um Negativ-Muster, die nicht vorhanden sein dürfen. Ein auf BPMN spezialisiertes Verfahren findet sich in GHOSE & KOLIADIS [17]. WEIDLICH ET AL. [36] verwenden stattdessen Log-Daten aus Informations-systemen in Kombination mit einem abstrakten Modell des Prozessverhaltens.

Die Identifikation von *Schwachstellen* zielt auf eine generelle Verbesserung der Geschäftsabläufe ab. Dies kann dadurch erreicht werden, dass entweder typische Schwachstellen oder Ansatzpunkte für zu implementierende Best Practices in den Modellen zu identifizieren sind. Solche Stellen sind zum Teil über Muster auffindbar. So erlaubt es beispielsweise der Ansatz von SMIRNOV ET AL. [31], sogenannte „Action Patterns“ in Prozessmodellen zu identifizieren. Dabei handelt es sich um Assoziationsregeln, die gemeinsam auftretende Aktivitäten miteinander verbinden. Ebenfalls auf Basis von Mustern lässt sich ein Verfahren entwickeln, das der Überprüfung von Prozessmodellen auf „soundness“ dient [12]. Soundness ist dabei ein notwendiges Kriterium, das die „Stimmigkeit“ eines Prozesses beschreibt (die umfasst bspw. Deadlockfreiheit). Ein ähnliches Verfahren findet sich in TOURÉ, BAÏNA & BENALI [32].

Da Größe sowie Anzahl betrieblicher Modelle in der Praxis immense Ausmaße annehmen können, ist *Komplexitätsmanagement* ein wichtiges Thema, um Prozessabläufe auf einem höheren Abstraktionsniveau zu beschreiben. Dies lässt sich durch das Auffinden häufig vorkommender Muster umsetzen, welche durch einen Verweis auf eine zentral abgelegte, abstrakte Beschreibung ersetzt werden. WEBER ET AL. [35] entwickeln dazu ein musterbasiertes Refactoring-Verfahren. Ähnliche Ansätze verfolgen REIJERS, MENDLING & DIJKMANN [27] sowie UBA ET AL. [33].

Ein weiteres Anwendungsfeld für Mustererkennung im Rahmen der Modellanalyse ist der *Vergleich* von Prozessmodellen. Hier werden zunächst alle möglichen Muster einer vorgegebenen Art aus zwei Modellen bestimmt, um im Anschluss daran die Schnittmenge der Muster zu berechnen, die besonders häufig in beiden Modellen auftreten. Dadurch ist eine Quantifizierung der Modellähnlichkeit möglich [37].

Ein weiteres, häufig auftretendes Problem ist das der *Modelltransformation*. Dabei soll ein gegebenes Modell in ein Modell einer anderen Modellierungssprache überführt werden. Übersetzungsregeln, die zur Umsetzung dieses Transformationsprozesses zu definieren sind, werden als abstrakte Muster repräsentiert, die es im Modell zu finden gilt. Verfahren zur Umsetzung der Modelltransformation stellen GARCÍA-BAÑUELOS [16] sowie OUYANG ET AL. [26] vor.

Betrachtet man die hier vorgestellte Literatur, so wird deutlich, dass die Suche nach strukturellen Mustern in Modellen ein Universalproblem ist, mit dem in Analyseverfahren

verschiedener Art umzugehen ist. Neben diesen auf der Mustersuche aufbauenden Methoden gibt es jedoch auch eine Reihe von Ansätzen, die ganz explizit dieses Problem – allerdings zugeschnitten auf spezielle Modellierungssprachen – adressieren. Zum Beispiel entwickeln AWAD & SAKR [2] ein Verfahren, der „Queries“ in BPMN-Modellen auswertet, also bestimmte Muster in ihnen findet. Vergleichbare Verfahren finden sich in BEERI ET AL. [5] sowie MOMOTKO & SUBIETA [23].

3 Grundlegende Definitionen und Problembeschreibung

Um die strukturelle Modellanalyse mit allgemeinen Verfahren der algorithmischen Graphentheorie zu unterstützen, ist zu untersuchen, ob die in der Wirtschaftsinformatik gängigen Modelle üblicherweise komplexitätsreduzierende Eigenschaften aufweisen. Dazu wird eine graphentheoretisch-formale Darstellungsweise von Modellen verwendet, die im Folgenden vorgestellt wird.

3.1 Konzeptionelle Modelle als Graphen

In diesem Beitrag werden konzeptionelle Modelle als ungerichtete, beschriftete Graphen $G = (V, E, \mathcal{L})$ mit einer Knotenmenge V, einer Kantenmenge $E \subseteq \{\{u, v\} | u, v \in V, u \neq v\}$ sowie einer Beschriftungsfunktion $\mathcal{L}: V \rightarrow L$ verstanden, welche jedem Knoten Bezeichner und Typ zuordnet. Da bestimmte Modelle (z. B. Prozessmodelle) jedoch gerichtete Graphen sind, werden entsprechende Kanten als Knoten interpretiert, die zwei Nachbarknoten sowie zwei Attribute „Quelle“ und „Ziel“ besitzen, die die Richtung der Kante bestimmen. Im Fall eines vollständig verbundenen Graphen werden so $n + n(n-1)/2 = (n^2 + n)/2$ Knoten dem ursprünglichen Modell hinzugefügt, was die angestrebte polynomielle Laufzeit der Suchalgorithmen nicht weiter beeinträchtigt.

3.2 Subgraphen und Subgraphisomorphie

Von dieser allgemeinen Modelldefinition ausgehend kann ein Subgraph $G' = (V', E', \mathcal{L}')$ eines Graphen G als ein Graph verstanden werden, dessen Knoten $V' \subseteq V$ und Kanten $E' \subseteq E$ Teilmengen des ursprünglichen Graphen sind. Zusätzlich muss die Beschriftungsfunktion $\mathcal{L}'$ für alle Knoten des Subgraphen gleich der ursprünglichen Beschriftungsfunktion $\mathcal{L}$ sein (d. h. $\mathcal{L}' = \mathcal{L}|_{V'}$). Von dieser Definition eines Subgraphen ausgehend, kann der Begriff Subgraphisomorphie wie folgt definiert werden: Formal heißt ein Mustergraph H subgraphisomorph zu einem Modellgraphen G, wenn eine Bijektion Φ zwischen den Knoten von H und einer Teilmenge der Knoten von G existiert, so dass für jede Kante (v,w) zwischen zwei Knoten v und w des Mustergraphen eine korrespondierende Kante $\{\Phi(v), \Phi(w)\}$ im Modellgraphen existiert.

4 Effiziente Mustersuche

Das Problem der Subgraphisomorphie ist im allgemeinen Fall NP-vollständig (vgl. [8]). Effiziente, polynomielle Algorithmen existieren, wenn angenommen wird, dass die zugrundeliegenden Graphen bestimmte Eigenschaften erfüllen. Im Folgenden werden diese Eigenschaften kurz vorgestellt.

4.1 Planarität

Ein Graph G wird planar genannt, falls es möglich ist, ihn auf einer Ebene zu zeichnen, ohne dass sich zwei Kanten schneiden. Eine formale Definition von Planarität wird durch das Robertson-Seymour-Theorem gegeben. Es besagt, dass ein Graph planar ist, falls er keinen vollständig verbundenen Graphen aus fünf Knoten (K^5) und keinen vollständig verbundenen bipartiten Graphen aus sechs Knoten ($K_{3,3}$) als Minor enthält [15]. Ein Minor H eines Graphen G beschreibt dabei einen isomorphen Subgraphen von G, der durch Kantenkontraktion entsteht. Bei einer Kantenkontraktion werden Kanten aus einem Graphen entfernt und die vormals verbundenen Knoten zu einem neuen Knoten verschmolzen.

In EPPSTEIN [14] und DORN [13] werden polynomielle Algorithmen für Subgraphisomorphie auf planaren Muster- und Modellgraphen vorgestellt. Ist $k = |V_H|$ die Anzahl der Knoten des Mustergraphen und $n = |V_G|$ die Anzahl der Knoten des Modellgraphen, ergibt sich eine Laufzeit von $2^{O(k)} \cdot n$ für planare Graphen. Die Algorithmen weisen somit eine in der Anzahl der Knoten des Modellgraphen lineare Laufzeitkomplexität auf.

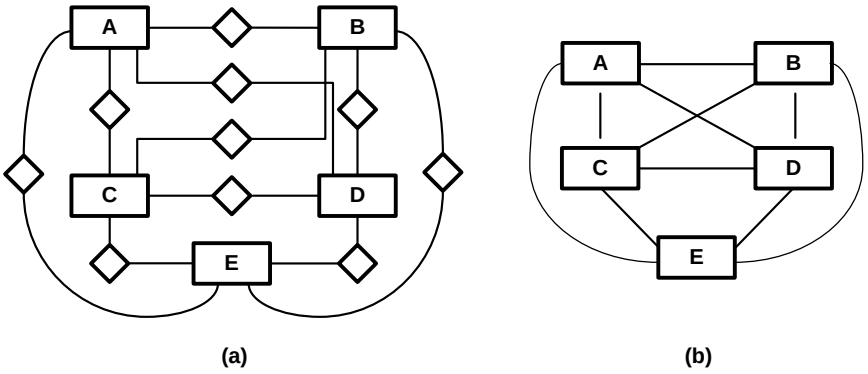

Bild 1: Beispiel für ein nicht-planares ERM (a), das K^5 (b) als Minor enthält

Aus Gründen der Lesbarkeit wird ein Modellierer darauf bedacht sein, nach Möglichkeit planare Modelle zu erstellen. Vorausgesetzt werden kann dies aber keinesfalls. Basierend auf den Meta-Modellen entsprechender Modellierungssprachen erlauben nahezu alle Sprachen die Erstellung von nicht-planaren Modellen. Ein Beispiel hierfür findet sich in Bild 1. Der Graph in 1(b) stellt einen Minor des Graphen 1(a) dar. Da der Graph 1(b) einem K^5 entspricht, ist dieses ERM nicht planar. Ähnliche nicht-planare Beispiele können für viele andere Sprachen wie z. B. EPK oder BPMN konstruiert werden.

4.2 Baumweite

Eine zweite komplexitätsreduzierende topologische Grapheigenschaft von Modellen ist die sogenannte *Baumweite*. Ein Graph G ist ein Baum, falls er verbunden ist und keine Zyklen besitzt. Viele Probleme können auf Bäumen effizient gelöst werden, unter anderem auch die Subgraphisomorphie. In SHAMIR & TSUR [29] wird ein effizienter Algorithmus für Subgraphisomorphie auf Bäumen vorgestellt. Der Algorithmus weist eine Laufzeit von $O(k^{1.5}/\log(k) \cdot n)$ für $k = |V_H|$ und $n = |V_G|$ auf, die somit linear in der Anzahl der Knoten des Modellgraphen ist. Es kann nicht davon ausgegangen werden, dass Modelle in jedem Fall Bäume sind. Dies zeigt sich beispielsweise in Prozessmodellen mit sich aufteilendem und wieder zusammenfließendem Kontrollfluss. Hierdurch entsteht ein Zyklus. Somit ist zu untersuchen, ob bzw. zu welchem Grad Modelle Bäume darstellen. Dies kann mit Hilfe des

Konzepts der Baumweite geschehen. Informal ausgedrückt misst die Baumweite, wie „baumähnlich“ ein Graph ist. Die Baumweite eines Graphen wird mit Hilfe der sogenannten Baumdekomposition berechnet. Eine Baumdekomposition eines Graphen $G = (V_G, E_G, \mathcal{L})$ ist ein Paar (T, W), das aus einem Baum $T = (V_T, E_T)$ und einem Mengensystem $W = (W_t : t \in V_T)$ besteht, so dass gilt:

- $\bigcup_{t \in V_T} W_t = V_G$
- $\forall \{u, v\} \in E_G$ gibt es ein W_t mit $u, v \in W_t$
- Wenn $t, t', t'' \in V_t$ und t' auf einem Pfad von t nach t'' liegen, dann gilt $W_t \cap W_{t''} \subseteq W_{t'}$

Die Baumweite einer Baumzerlegung ergibt sich aus der maximalen Anzahl an Knoten in einer der Mengen W_t minus 1, d. h. $\max(|W_t| - 1 | \, t \in V_t)$. Die Baumweite tw von G ist nun die kleinste Weite aller möglichen Baumzerlegungen von G [28].

Es existieren effiziente Algorithmen zur Subgraphisomorphie, die Eigenschaften von Graphen mit kleiner, begrenzter Baumweite ausnutzen. Der Ansatz von ALON, YUSTER & ZWICK [1] weist eine Komplexität von 2^(O(k))·n^(tw+1) auf, wobei k die Knotenzahl des Mustergraphen H, n die Anzahl Knoten des Modellgraphen G und tw die Baumweite des Mustergraphen H repräsentieren. Die Komplexität ist somit polynomiell in n mit einem Exponenten der von der Baumweite tw des gesuchten Graphen abhängig ist.

Der Algorithmus zur Suche eines Subgraphen kann verbessert werden, indem eine weitere Einschränkung des Musters H gefordert wird. Der Grad eines Knoten ist definiert als die Anzahl von Kanten, die mit ihm verbunden sind. $\Delta(H)$ sei der größte Grad aller Knoten des Mustergraphen. Der Algorithmus von MATOUŠEK & THOMAS [21] besitzt eine Komplexität von $O(k^{tw+1} \cdot n)$ und setzt voraus, dass $\Delta(H)$ durch einen maximalen Grad Δ begrenzt ist und sowohl G als auch H eine maximale Baumweite von tw einhalten. Der Algorithmus ist daher sogar polynomiell in der Anzahl Knoten des (üblicherweise relativ kleinen) Mustergraphen und nicht mehr in der Anzahl Knoten des Modellgraphen.

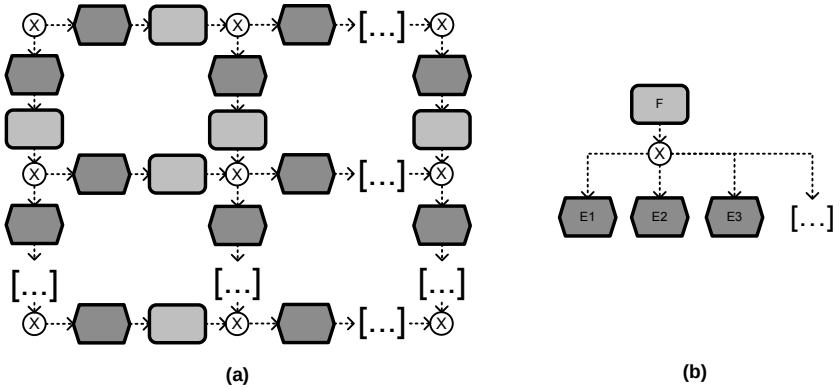

Bild 2: Unbegrenzte Baumweite (a) und unbegrenzter Knotengrad (b) bei EPKs

Wie zuvor bei der Planarität kann auch über die Baumweite von Modellen allein auf Basis der Modellierungssprache keine Aussage gemacht werden. Ein Beispiel für eine theoretisch unbeschränkte Baumweite wird durch das EPK-Modell in Bild 2 (a) aufgezeigt. Theoretisch ist es möglich, unendlich große Gitter aus Ereignissen, Funktionen und Konnektoren zu konstruieren. Die Baumweite eines Gitters entspricht seiner Höhe und Breite. Daher ist die

Baumweite einer EPK theoretisch unbegrenzt. Zudem kann eine unbegrenzte Anzahl an Ereignissen mit einem Konnektor verbunden werden (siehe Bild 2 (b)). Der maximale Knotengrad eines EPK-Modells ist daher theoretisch auch unbegrenzt. Ähnliche Beispiele können auch für andere Modellierungssprachen konstruiert werde. So kann in BPMN ein Gitter durch „join“ und „split“ Konnektoren erzeugt werden. Darüber hinaus können Aktivitäten mit beliebig vielen Ressourcen annotiert werden. Ähnliche Beispiele können für viele andere Sprachen wie z. B. ERM oder Klassendiagramme konstruiert werden.

Diese Ausführungen verdeutlichen, dass für jede graphentheoretische Eigenschaft, die eine effiziente Mustersuche ermöglicht, Beispielmodelle konstruiert werden können, die diese Eigenschaft verletzen. Um die Anwendbarkeit der entsprechenden Algorithmen zu demonstrieren, ist somit eine empirische Analyse einer großen Menge konzeptioneller Modelle durchzuführen. Kann gezeigt werden, dass die Mehrheit dieser Modelle entsprechende Eigenschaften aufweisen, so können die oben vorgestellten effizienten Verfahren zur Mustererkennung in bestehende Analyseansätze integriert werden.

5 Aufbau der empirischen Analyse

Die Modelldatenbank, auf der die Analyse aufsetzt, besteht aus Modellen von fünf verschiedenen Modellierungsprojekten. Die ersten beiden Projekte (ÖV1 und ÖV2) entstanden im Rahmen von Prozessmodellierungsprojekten in der öffentlichen Verwaltung. Sie enthalten 2164 (ÖV1) bzw. 604 (ÖV2) EPK-Modelle. Das größte Modell der ÖV1-Datenbank enthält 912 Elemente. Die dritte Quelle für EPK- und ERM-Modelle enthält das von BECKER und SCHÜTTE vorgestellte Referenzmodell für den Handelssektor (Handel) [3]. Die Datenbank enthält insgesamt 35 ERMs und 54 EPKs. Die vierte Datenbank entstand im Rahmen eines Meta-Modellierungsprojektes (MM) und besteht aus 33 ERM-Modellen. Die fünfte Quelle enthält 26 UML-Klassendiagramme (UML KD) (UML [24]) aus dem Common Warehouse Metamodel (CWM [25]). Insgesamt werden damit 2822 EPK- und 68 ERM-Modelle sowie 26 UML-Klassendiagramme analysiert. Alle Modelle wurden in eine XML-Datenbank überführt, die auf dem in Abschnitt 3.1 beschriebenen Graphschema basiert.

Um die Planarität der Modelle zu untersuchen, wurde die Boost graph library (vgl. [30]) genutzt. Die Bibliothek implementiert den Boyer-Myrvold-Planaritätstest, welcher eine Komplexität von $O(n)$ aufweist. Zur Berechnung der Baumweite wurde LibTW (vgl. [9]) eingesetzt. Die Bibliothek stellt verschiedene Algorithmen zur Berechnung von exakter Baumweite sowie deren oberer und unterer Schranke zur Verfügung. Da der Algorithmus zur exakten Berechnung der Baumweite für Modelle mit mehr als 100 Knoten fehlerhaft arbeitet, wurden obere und untere Schranke bestimmt. Die untere Schranke wurde mit Hilfe der Algorithmen MaximumMinimumDegreePlusLeastC und MinorMinWidth ermittelt, während die obere Schranke durch GreedyFillIn und GreedyDegree berechnet wurden. Falls obere und untere Schranke übereinstimmen, kann die Baumweite aus den Schranken geschlossen werden. Dies war für alle analysierten Modelle der Fall. Die Untersuchung wurde auf einem Dell XPS 1530M Notebook mit Windows 7 Service Pack 1 durchgeführt. Das Notebook enthält eine Intel Core Duo T7250 CPU mit 2 GHz und 2048 GB Arbeitsspeicher. Die Berechnung aller relevanten Kennzahlen dauerte bei Modellen mit höchstens 100 Knoten zwischen fünf Millisekunden und einer Sekunde, bei Modellen mit mehr als 100 Knoten bis zu drei Sekunden.

6 Empirische Analyse

6.1 Planarität

Tabelle 1 fasst die Ergebnisse des Tests auf Planarität in der Modelldatenbank zusammen. Trotz der Einschränkung, dass konzeptuelle Modelle generell nicht-planar sein können, zeigt sich, dass fast alle Modelle planar sind. Mit 90.9% enthält die MM-Datenbank die wenigsten planaren Modelle. Für alle anderen Datenbanken ist der Anteil größer als 98%. Wir erklären diese Abweichung dadurch, dass im Rahmen des Metamodellierungsprojekts vergleichsweise große Modelle erstellt wurden, mit denen zudem vergleichsweise komplexe Sachverhalte zur Spezifikation konfigurierbarer Modellierungsmethoden konstruiert wurden.

Projekt	Planare EPKs		Planare ERMs		Planare UML KDs	
	Absolut	Prozent	Absolut	Prozent	Absolut	Prozent
ÖV1	2127	98,3%	—	—	—	—
ÖV2	602	99,6%	—	—	—	—
Handel	53	98,1%	35	100%	—	—
MM	—	—	30	90,9%	—	—
CWM	—	—	—	—	26	100%
Total	2782	98,6%	65	95,6%	26	100%

Tabelle 1: Planarität

Das Analyseergebnis zeigt, dass die Mehrheit der hier untersuchten Modelle durch effiziente Algorithmen, die die Planarität von Graphen ausnutzen, analysiert werden kann. Da es möglich ist, Planarität in linearer Laufzeit zu überprüfen (vgl. [7]), kann diese Eigenschaft effizient getestet und danach ein entsprechender Mustersuche-Algorithmus angewandt werden, dessen Laufzeit linear in der Anzahl der Knoten des Modellgraphen ist. Musterentsprechungen können daher in linearer Zeit für fast alle Modelle der vorliegenden Datenbanken berechnet werden. Es wird vermutet, dass die Planarität der großen Mehrheit der Modelle in der Tatsache begründet ist, dass diese nicht maschinell erstellt, sondern von Menschen entwickelt werden, die die in den Modellen ausgedrückten Sachverhalte unbewusst einfach darstellen. Dies verringert die strukturelle Komplexität der resultierenden Graphen. Prozessmodelle enthalten meist lineare Abfolgen von Aktivitäten, die nicht durch übermäßig komplizierte Verzweigungsstrukturen unterbrochen werden. Auch Datenmodelle und Klassendiagramme sind im Wesentlichen dünn besetzte Graphen, in denen jeder Knoten mit einer begrenzten Anzahl an Kanten verbunden ist. Dies begünstigt ihre planare Topologie.

6.2 Baumweite

Tabelle 2 präsentiert die Ergebnisse bezüglich der Baumweite der untersuchten Modelle. Die maximal ermittelte Baumweite beträgt vier. Fast 30 Prozent der EPK-Modelle weisen eine Baumweite von eins auf und sind damit Bäume. Für UML ist dies sogar für mehr als 65% der Modelle gegeben. Jedoch sind ERMs mit einem Anteil von 13,2% eher seltener Bäume. Allerdings weisen fast alle weiteren Modelle eine Baumweite von zwei oder drei auf. Nur sehr wenige Modelle besitzen eine Baumweite von vier.

Soll ein Subgraph mit einem die Baumweite ausnutzenden Algorithmus gesucht werden, wird dazu eine Baumzerlegung mit geringer Baumweite benötigt. Die Untersuchung zeigt, dass

Heuristiken zur Abschätzung der Baumweite nach oben gute Ergebnisse liefern. Da sie versuchen, heuristisch eine möglichst gute Baumzerlegung zu konstruieren, ist nach ihrer Anwendung die Baumzerlegung direkt vorhanden. So kann die Mustersuche in polynomieller Laufzeit erfolgen (vgl. [6]). Auch diese Ergebnisse sind vor dem Hintergrund interpretierbar, dass konzeptionelle Modelle von Menschen entwickelt werden. Die linearen Ablaufbeschreibungen von Prozessmodellen führen zu Bäumen oder baumähnlichen Graphen, die von entsprechenden Algorithmen einfach zu analysieren sind. Im Gegensatz zu den Ergebnissen aus Abschnitt 6.1 besteht ein Unterschied zwischen Prozess- und Datenmodellen. Letztere repräsentierten eher Graphen mit höherer Baumweite. Die Mehrheit der untersuchten Modelle weist eine Baumweite von zwei oder drei auf. Auch die Anzahl der Datenmodelle mit Baumweite vier ist deutlich höher als bei Prozessmodellen. Aus Erfahrungen vermuten wir, dass dies durch in Datenmodellen häufiger auftretenden zyklischen Strukturen begründet ist.

Projekt	$tw = 1$		$tw = 2$		$tw = 3$		$tw = 4$	
	Absolut	Prozent	Absolut	Prozent	Absolut	Prozent	Absolut	Prozent
EPK Gesamt	908	32,2%	1619	57,4%	281	10%	14	0,5%
ÖV1	770	35,6%	1143	52,8%	238	11%	13	0,7%
ÖV2	134	22,2%	430	71,2%	39	6,5%	1	0,2%
Handel	4	7,4%	46	85,2%	4	7,4%	0	0%
ERM Gesamt	9	13,2%	42	61,8%	15	22,1%	2	2,9%
Handel	4	11,4%	29	82,9%	2	5,7%	0	0%
MM	5	15,2%	13	39,4%	13	39,4%	2	6,1%
UML KD Total (CWM)	17	65,4%	8	30,8%	1	3,8%	0	0%
Total	934	32%	1669	57,2%	297	10,2%	16	0,5%

Tabelle 2: Baumweite

6.3 Knotengrad

In Tabelle 3 sind die Messungen des maximalen Knotengrades $\Delta(G)$ der Modelle zusammengefasst. Im Mittel schwanken die Werte zwischen eins und fünf. Mehr als 85% aller Modelle haben ein $\Delta(G)$ kleiner 15. Im Vergleich der Modellierungssprachen finden sich die höchsten Werte bei den EPKs der ÖV1-Datenbank. Die Mehrheit der untersuchten Datenmodelle weist einen Knotengrad auf, der zwischen eins und zehn liegt. Bei Prozessmodellen treten jedoch auch höhere Knotengrade auf. Es wird angenommen, dass ein häufig verwendetes Konstrukt in Prozessmodellen – die multiple Verzweigung mithilfe eines Konnektors – für diesen hohen maximalen Knotengrad verantwortlich ist.

Die Ergebnisse demonstrieren jedoch, dass die meisten analysierten Modelle keine hohen Knotengrade aufweisen. Folglich können Graphen von Strukturmustern in den meisten Fällen auch kein hohes $\Delta(H)$ aufweisen. Dies zeigt, dass die oben vorgestellte Erweiterung des auf der Baumweite basierenden Algorithmus in vielen Fällen angewendet werden kann.

Grad / Projekt	1-5	6-10	11-15	16-20	21-25	>25
EPK Gesamt	1235 (43.2%)	774 (27%)	463 (16.2%)	217 (7.6%)	97 (3.3%)	72 (2.5%)
ÖV1	716 (32.6%)	647 (29.4%)	451 (20.5%)	217 (9.9%)	97 (4.4%)	72 (3.3%)
ÖV2	468 (77.5%)	124 (20.5%)	12 (2%)	0 (0%)	0 (0%)	0 (0%)
Handel	51 (94.4%)	3 (5.6%)	0 (0%)	0 (0%)	0 (0%)	0 (0%)
ERM Gesamt	41 (60%)	26 (38%)	0 (0%)	1 (1%)	0 (0%)	0 (0%)
Handel	16 (45.7%)	19 (54.3%)	0 (0%)	0 (0%)	0 (0%)	0 (0%)
MM	25 (75.8%)	7 (21.2%)	0 (0%)	1 (3%)	0 (0%)	0 (0%)
UML Total (CWM)	8 (27%)	5 (17%)	10 (33%)	7 (23%)	0 (0%)	0 (0%)
Gesamt	1284 (43.4%)	805 (27.2%)	473 (16%)	225 (7.6%)	97 (3.3%)	72 (2.4%)

Tabelle 3: Maximale Knotengrade

7 Fazit & Ausblick

Die hier vorgestellte Analyse verdeutlicht, dass die untersuchten Modelle Eigenschaften aufweisen, die eine effiziente Mustersuche auf Basis graphentheoretischer Algorithmen erlauben. Die Mehrheit der hier untersuchten Modelle weist eine begrenzte Baumweite auf und ist planar. Weniger als 2% der Modelle erfüllen letztgenannte Eigenschaft nicht. Ferner konnte kein Modell mit einer Baumweite größer als vier identifiziert werden. Lediglich 11% der Modelle besitzen eine Baumweite von drei oder vier. Der beobachtete Knotengrad liegt bei der Mehrheit der Modelle zwischen eins und fünf.

Diese Ergebnisse legen nahe, dass effiziente, allgemein anwendbare Verfahren der Mustererkennung in bestehende Modellanalyseansätze integriert werden können. Ein entscheidender Vorteil der Nutzung eines allgemeinen Verfahrens ist dessen Transferfähigkeit. Es kann in Analyseansätzen verwendet werden, die für beliebige Modellierungssprachen und Analysezwecke (Modellvergleich, Compliance-Überprüfung, etc.) konzipiert sind. Neben der Transferfähigkeit ermöglicht die Effizienz eines entsprechenden Ansatzes die breite Anwendung, selbst wenn es sich bei den zu analysierenden Modellen um sehr große Modelle (bspw. Prozesslandschaften) handelt – und gerade im Rahmen der Analyse sehr umfangreicher Modelle ist eine automatisierte Analyseunterstützung besonders wichtig.

Dieser Beitrag stellt die empirische Grundlegung für die Entwicklung einer generischen und effizienten Modellanalysemethode zur Verfügung. Diese Methode wird ein gegebenes Modell in zwei Phasen analysieren. In einem ersten Schritt werden die graphentheoretischen Eigenschaften des Modells bestimmt. In einem nächsten Schritt wird der entsprechend effizienteste Algorithmus für die Mustersuche ausgewählt und angestoßen. Weitere Forschung wird sich dem Ziel widmen, diese generische Analysemethode zu implementieren und ihre Effizienz nicht nur theoretisch, sondern auch empirisch zu bestimmen. Es ist anzunehmen, dass nicht für jede Fragestellung, die im Rahmen der Modellanalyse auftritt, Verfahren der exakten Mustersuche angewendet werden können. Zu diesem Zweck sind weiterhin Algorithmen zu untersuchen, die zu einem vorgegebenen Muster ähnliche Subgraphen identifizieren (Subgraphhomöomorphie; vgl. z. B. [18]). Da die syntaktischen, semantischen und pragmatischen Freiheitsgrade der Modellierung durch die hier thematisierten Modelleigenschaften stark beeinflusst werden, konzentrieren sich langfristige

Forschungsaktivitäten darauf, konkrete Modellierungsrichtlinien zu formulieren und ggf. als Modellierungsanleitungsverfahten zu implementierten, um bereits von vornherein eine effiziente Analysierbarkeit der Modelle zu begünstigen.

8 Literatur

[1] Alon, N; Yuster, R; Zwick, U (1995): Color-coding. Journal of the ACM 42(4):844-856.

[2] Awad, A; Sakr, S (2010): Querying Graph-Based Repositories of Business Process Models. In Yoshikawa, M; Meng, X; Yumoto, T; Ma, Q; Sun, L; Watanabe, C (Hrsg.), Database Systems for Advanced Applications. Springer, Berlin / Heidelberg, 33-44.

[3] Becker, J; Schütte, R (2004): Handelsinformationssysteme. 2. Auflage. Redline Wirtschaft bei Verlag Moderne Industrie, Landsberg.

[4] Becker, J; Weiß, B; Winkelmann, A (2011): Automatic Identification of Structural Process Weaknesses – Experiences with Semantic Business Process Modeling in the Financial Sector. In: Schwabe, ABG (Hrsg.), Proc. of the 10th Int. Conf. on Wirtschaftsinformatik. Zürich, Switzerland.

[5] Beeri, C; Eyal, A; Kamenkovich, S; Milo, T (2008): Querying business processes with BP-QL. Information Systems Journal 33(6):477-507.

[6] Bodlaender, HL (2005): Discovering Treewidth. SOFSEM 2005 Theory and Practice of Computer 306:1-16.

[7] Booth, K; Lueker, G (1976): Testing for the Consecutive Ones Property, Interval Graphs, and Graph Planarity Using PQ-Tree Algorithms. Journal of Computer System Sciences 13(13):335-379.

[8] Cook, SA (1971): The complexity of theorem-proving procedures. In: Proc. of the 3rd annual ACM symposium on Theory of computing. New York, USA.

[9] Dijk, T van; Heuvel, J-P van den; Slob, W (2011): LibTW. http://www.treewidth.com/.

[10] Dijkman, R; Dumas, M; García-Bañuelos, L (2009): Graph Matching Algorithms for Business Process Model Similarity Search. In Dayal, U; Eder, J; Koehler, J; Reijers, H (Hrsg.), Business Process Management. Springer Berlin / Heidelberg, 48-63.

[11] Dongen, BF Van; Dijkman, R; Mendling, J (2008): Measuring similarity between business process models. In Bellahsène, Z; Léonard, M (Hrsg.), Advanced Information Systems Engineering. Springer, Berlin / Heidelberg, 450-464.

[12] Dongen, BF van; Mendling, J; Aalst, WMP van der (2006): Structural Patterns for Soundness of Business Process Models. Proc. of the 10th IEEE International Enterprise Distributed Object Computing Conference :116-128.

[13] Dorn, F (2010): Planar Subgraph Isomorphism Revisited. In: Proc. of the 27th Int. Symposium on Theoretical Aspects of Computer Science. Nancy, France.

[14] Eppstein, D (1999): Subgraph Isomorphism in Planar Graphs and Related Problems. Journal of Graph Algorithms and Applications 3(3):1-27.

[15] Fellows, MR (1989): The Robertson-Seymour theorems: A survey of applications. Contemporary Mathematics 89:1-18.

[16] García-Bañuelos, L (2008): Pattern Identification and Classification in the Translation from BPMN to BPEL. In Meersman, R; Tari, Z (Hrsg.), On the Move to Meaningful Internet Systems: OTM 2008. Springer-Verlag, Berlin / Heidelberg, 436-444.

[17] Ghose, A; Koliadis, G (2007): Auditing Business Process Compliance. In Krämer, B; Lin, K-J; Narasimhan, P (Hrsg.), Service-Oriented Computing – ICSOC 2007. Springer, Berlin / Heidelberg, 169-180.

[18] Gupta, A; Nishimura, N (1994): Sequential and parallel algorithms for embedding problems on classes of partial k-trees. In Schmidt, E; Skyum, S (Hrsg.), Algorithm Theory – SWAT '94. Springer, Berlin / Heidelberg, 172-182.

[19] Hajiaghayi, M; Nishimura, N (2007): Subgraph isomorphism, log-bounded fragmentation, and graphs of (locally) bounded treewidth. Journal of Computer and System Sciences 73(5):755-768.

[20] Knuplesch, D; Ly, L; Rinderle-Ma, S; Pfeifer, H; Dadam, P (2010): On Enabling Data-Aware Compliance Checking of Business Process Models. In Parsons, J; Saeki, M; Shoval, P; Woo, C; Wand, Y (Hrsg.), Conceptual Modeling – ER 2010. Springer, Berlin / Heidelberg, 332-346.

[21] Matoušek, J; Thomas, R (1992): On the complexity of finding iso- and other morphisms for partial k-trees. Discrete Mathematics 108(1-3):343-364.

[22] Mendling, J (2007): Detection and prediction of errors in EPC business process models. Wien.

[23] Momotko, M; Subieta, K (2004): Process Query Language : A Way to Make Workflow Processes More Flexible. Advances in Databases and Information Systems 54:306-321.

[24] Object Management Group (2011): Unified Modeling Language. http://uml.org/.

[25] Object Management Group (2011): Common Warehouse Metamodel 1.1. http://www.omg.org/spec/CWM/1.1/.

[26] Ouyang, C; Dumas, M; Hofstede, AHMT; Aalst, WMP van der (2008): Pattern-based Translation of BPMN Process Models to BPEL Web Services. International Journal of Web Services Research 5(1):1-21.

[27] Reijers, HA; Mendling, J; Dijkman, RM (2011): Human and automatic modularizations of process models to enhance their comprehension. Information Systems Journal 36(5):881-897.

[28] Robertson, N; Seymour, P (1984): Graph minors. III. Planar tree-width. Journal of Combinatorial Theory, Series B 36(1):49-64.

[29] Shamir, R; Tsur, D (1999): Faster Subtree Isomorphism. Journal of Algorithms 33(2):267-280.

[30] Siek, J; Lee, LQ; Lumsdaine, A (2002): The Boost Graph Library: User Guide and Reference Manual. Addison-Wesley, Boston.

[31] Smirnov, S; Weidlich, M; Mendling, J; Weske, M (2009): Action Patterns in Business Process Models. In Baresi, L; Chi, C-H; Suzuki, J (Hrsg.), Service-Oriented Computing. Springer, Berlin / Heidelberg, 115-129.

[32] Touré, F; Baïna, K; Benali, K (2008): An Efficient Algorithm for Workflow Graph Structural Verification. In Meersmann, R; Tari, Z (Hrsg.), On the Move to Meaningful Internet Systems: OTM 2008. Springer, Berlin / Heidelberg, 392-408.

[33] Uba, R; Dumas, M; Garcia-Banuelos, L; Rosa, M La (2011): Clone Detection in Repositories of Business Process Models. Brisbane.

[34] Ullmann, JR (1976): An Algorithm for Subgraph Isomorphism. Journal of the ACM 23(1):31-42.

[35] Weber, B; Reichert, M; Mendling, J; Reijers, HA (2011): Refactoring large process model repositories. Computers in Industry 62(5):467-486.

[36] Weidlich, M; Polyvyanyy, A; Desai, N; Mendling, J (2010): Process Compliance Measurement Based on Behavioural Profiles. In Pernici, B (Hrsg.), Advanced Information Systems Engineering. Springer, Berlin / Heidelberg, 499-514.

[37] Yan, Z; Dijkman, R; Grefen, P (2010): Fast Business Process Similarity Search with Feature-Based Similarity Estimation. In Meersmann, R; Dillon, T; Herrero, P (Hrsg.), On the Move to Meaningful Internet Systems: OTM 2010. Springer, Berlin / Heidelberg, 60-77.